REMARKS

Claims 1-4, 8-17 and 21-28 are pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Allowable Subject Matter:

Claims 5-7 and 18-20 were objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form. Applicants thank the Examiner for consideration of these claims, but assert that claims 5-7 and 18-20 are allowable in their present form.

Section 102(e) Rejection:

The Office Action rejected claims 1-4, 8-17 and 21-28 under 35 U.S.C. § 102(e) as being anticipated Weschler (U.S. Patent 6,842,903). Applicants respectfully traverse this rejection for at least the reasons below.

Regarding claim 1, contrary to the Examiner assertion, **Weschler fails to disclose receiving an address for a service within the distributed computing environment and linking the address to a pre-generated message interface for accessing the service, wherein the pre-generated message interface is implemented by computer-executable code installed on said device during a code-build process for installation of code on the device to implement said receiving and said linking**. As noted previously, Weschler teaches a method for providing dynamic references between services in a computer system that allows one service to obtain a reference to another service without requiring specific knowledge of the other service. In Weschler's system, pluggable modules may be used to provide additional functionality for a service and a service connector interface may encapsulate the logic to locate an instance of a particular service.

The Examiner relies on col. 8, lines 5-10 of Weschler describing pluggable interfaces and pluggable modules that provide supporting functionality and implement program behavior for core profiling engine 201. Weschler describes that modules are "plugged in by specifying an initialization parameter" that "comprises an address or fully qualified path pointing to a location *at which the plug-in module is stored*" (emphasis added). The address used by Weschler to plug-in a module is the address from which a plug-in module is loaded.

Firstly, Weschler's plugging in of a module cannot be considered the same as, nor does it disclose, linking an address for a service to a pre-generated message interface, as recited in Applicants' claim. For instance, Applicants' claim requires that a pre-generated message interface installed on the device during a code-build process for installation of code on the device to implement said receiving and said linking. However, Weschler's plug-in module is specifically downloaded and installed at runtime, not during a code-build process for installation of code to implement said receiving and said linking. In Weschler's system, previously installed core profile engine 201 implements the downloading and installing of plug-in modules (Weschler, FIG. 2 and col. 6, lines 24-53).

Thus, Weschler's plug-in module is downloaded and installed in a completely separate process from any code-build process for installation of code on the device to implement said receiving and said linking. Clearly, in contrast to the Examiner contention, Weschler's plug-in modules cannot be considered the same as, nor do they disclose, the specific limitation of *linking an address for a service to a pre-generated message interface for accessing the service, where the pre-generated message interface is implemented by computer-executable code installed on the device during a code-build process for installation of code on the device to implement said receiving and said linking*, as recited in Applicants' claim.

In the Response to Arguments, the Examiner argues, "Weschler disclosed … a mechanism through which the application can obtain a reference (the URL address) to the service and use it" and further contends, "Where the plug-in is used to generate an interface to the service, and the application casts to the interface" (Office Action, pp. 8-9). However, Applicants are not arguing that Weschler's system does not communicate with or otherwise use a service. The Examiner's response fails to address Applicants' argument that in Weschler's system, as pointed out and relied on by the Examiner, the client device downloads a plug-module that then provides the additional functionality, including accessing a service. However, a service address in Weschler's is not linked to a pre-generated message interface as recited in Applicants' claim (e.g., *a pre-generated message interface implemented by computer-executable code installed on the device during a code-build process for installation of code on the device to implement said receiving and said linking*).

Additionally, the Examiner erroneously contends that since Weschler's plug-in module is associated with the address, Weschler's system includes linking an address to a pre-generated message interface (Office Action, pp. 8-9). Applicants have previously pointed out that, by definition, "casting to an interface" is not the same as, nor does it disclose, "linking an address to a pre-generated message interface" and that casting refers to converting a programming entity (e.g. a variable, parameter, object, etc.) from one programming type to another. Indeed, "casting" as used by Weschler has nothing to do with linking an address to a pre-generated message interface for accessing a service.

In response, the Examiner asserts, "While the Applicant presents the dictionary definition of 'casting' it would be obvious to a person of ordinary skill in the art that in the cited portion Weschler does not refer to how different types of parameters/variables are involved" and that "the dictionary meaning of 'casting' would be totally out of context in the cited portion of Weschler." The Examiner further states, "In the cited portions Weschler disclosed how a client application prepares to access a service using an interface." However, the Examiner fails to take into account that, as is well known in the art, "casting", as part of a client application preparing to access a service using an interface, normally and routinely involves converting parameters/variables from one programming type to another.

Moreover, it is clearly improper for the Examiner to rely on an obviousness standard in a rejection under 35 U.S.C. §102. Applicants respectfully remind the Examiner that anticipation requires the presence in a single prior art reference disclosure of <u>each and every limitation</u> of the claimed invention, <u>arranged as in the claim</u>. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The <u>**identical**</u> **invention** <u>**must**</u> be shown <u>in as complete detail</u> as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). The Examiner's reliance on it being "obvious to a person of ordinary skill in the art" in the rejection of claim 1 is clearly improper for a rejection based on anticipation.

The Examiner also notes, "that code generation, linking and binding are well-known steps in the application development process and are required for enabling the application to interact with the operating system" and that "when Weschler disclosed 'casting to an interface' it would be obvious that Weschler was referring to an execution preparation/instantiation process including code generation, linking, and binding (Office Action, p. 9). Applicants respectfully disagree. While "code generation, linking and binding" may well known in other contexts, Applicants' claim does not recite mere code generation, linking and binding. Weschler's system does not include or disclose <u>linking the address to a pre-generated message interface for accessing the service</u>, wherein the <u>pre-generated</u> message interface is implemented by computer-executable code <u>installed on</u> said device <u>during a code-build process for installation of code on the device to implement said receiving and said linking</u>. The fact that, in other contexts, code generation, linking and binding may be well known does not alter the clearly conclusion that Weschler does not ***disclose*** the <u>specific limitations</u> of Applicants' claim.

Additionally, the Examiner is again improperly arguing that the subject matter of Applicants' claim is *obvious* in view of Weschler. Claim 1 is rejected under the anticipation standard, not the obviousness standard.

Moreover, the Examiner is incorrect that it would be obvious, in view of Weschler, to perform linking the address to a pre-generated message interface for accessing the service, wherein the pre-generated message interface is implemented by computer-executable code installed on said device during a code-build process for installation of code on the device to implement said receiving and said linking, as recited in Applicants' claim. There is no suggestion whatsoever of these specific limitations in Weschler.

As discussed above, Weschler fails to disclose receiving an address for a service within the distributed computing environment and linking the address to a pre-generated message interface for accessing the service, wherein the pre-generated message interface is implemented by computer-executable code installed on said device during a code-build process for installation of code on the device to implement said receiving and said linking. Therefore, Weschler cannot be said to ***anticipate*** claim 1.

Regarding claim 2, Weschler fails to disclose a message interface of the message endpoint **verifying that the messages sent to the service comply with a message schema for the service.** The Examiner cited column 6, lines 25-50 of Weschler. However, the cited passage states that Weschler's core profile engine 201 provides a limited set of functions includes, among others, "management utilities for defining schemas." Weschler does not describe verifying that messages sent to the service comply with a message schema for the service. Furthermore, Weschler does not state that a schema managed via the functions provided by the core profile engine 201 is usable to verify messages sent to the service. Instead, Weschler teaches that core profile engine 201 provides management utilities for *defining* schemas. The fact that Weschler's core profile engine 201 provides functions for defining schemas has nothing do with, and fails to disclose, verifying that messages sent to a service comply with a message schema for the service.

Applicants again assert that a single mention of "management utilities for defining schemas", as cited by the Examiner, does not in any way disclose the specific limitation of verifying that message sent to a service comply with a message schema for the service.

In the Response to Arguments, the Examiner cites Weschler in disclosing that the response message is sent back through API 203 to the appropriate protocol adapter 205 (or built-in adapter 205) to the requesting client application 202 (column 7, lines 18-20.) The Examiner contends that Weschler disclosed the verification limitation of claim 2 "because determining the appropriate protocol adapter would have inherently included verification for compliance with the message schema for the service." Applicants respectfully disagree. First, there is nothing in Weschler that describes <u>determining</u> the appropriate protocol, or how such a determination may be made. The only reference to setting a protocol when exchanging messages is found in column 9, lines 13-14, "Because the service 306 provider also provides and programs the service connector 304, the protocol for effectuating this is set at that time." This passage clearly does not disclose that the protocol is determined from a message itself, as the Examiner seems to imply, or that a message is (or should be) analyzed to see if it complies with a particular protocol (or schema) for this or any other reason. In fact, since the response message is sent from core engine 201 and may be generated by core engine 201 to be correct (i.e., compliant to the same protocol used for the request message) by construction. Applicants assert that none of the features of the system disclosed in Weschler necessarily <u>require</u> verification of compliance to a message schema.

"To establish inherency, the extrinsic evidence 'must make clear that the missing descriptive matter is **<u>necessarily present</u>** in the things described in the reference, and that it would be so recognized by persons of ordinary skill. Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient.'" *In re Robertson*, 169 F.3d 743, 745, 49 USPA2d 1949, 1950-51 (Fed. Cir. 1999) (emphasis added). Applicants assert that it would clearly not be <u>necessary</u> in the system of Weschler to <u>verify compliance</u> with a particular message schema for a given service in order to determine an appropriate protocol adapter, to send a response message to an appropriate protocol adapter, or to perform any other functions described in Weschler.

For at least the reasons above, the rejection of claim 2 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 9, 15, and 22.

Regarding claim 8, Weschler fails to disclose **receiving a schema defining messages for accessing the service; generating message endpoint code according to the schema; and linking said message endpoint code into executable operating code for the device and installing the message endpoint code and operating code onto the device**. In the previous Response, Applicants noted that Weschler teaches a method for providing dynamic references between services in a computer system that allows one service to obtain a reference to another service without requiring specific knowledge of the other service and that Weschler teaches a service connector interface that encapsulates the logic necessary to locate an instance of a particular service.

In the Response to Arguments, the Examiner cites column 6, lines 30-35 "gaining a reference to data store adapters" as teaching *receiving a schema defining messages for accessing the service*. The Examiner also cites column 9, lines 20-25 "the application casts to the interface" and column 8, lines 60-65 "service connector may be compiled along with the application" as teaching *generating message endpoint code according to said schema* and *linking said message endpoint code into executable operating code for the device*. The Examiner submits that data store adapters would inherently involve a schema for accessing the data structures involved. First, Applicants note that the Examiner's citation is column 6 describes gaining references to plug-in services, some of which may be data store adapters. Even if these data store adapters involved a schema, there is nothing in Weschler that describes receiving a schema defining messages for accessing the data store, as recited in claim 8. Furthermore, the Examiner's references to casting to an interface and compiling a service connector also do not teach generating code (i.e., *generating message endpoint code*) according to the schema, as recited in claim 8.

Applicants also note that Weschler does not teach *installing the message endpoint code* (i.e., message endpoint code generated according to the message schema) *and operating code onto the device*, nor does Weschler teach this limitation of claim 8.

Thus, for at least the reasons presented above, the rejection of claim 8 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claim 14 and 27.

Section 103(a) Rejection:

The Examiner rejected claims 1-4, 8-17 and 21-28 under 35 U.S.C. § 103(a) as being unpatentable over Roberts et al. (U.S. Patent 6,560,633) (hereinafter "Roberts") in view of Chen et al. (U.S. Publication 2002/0062334) (hereinafter "Chen"). Applicants respectfully traverse this rejection for at least the following reasons.

Regarding claim 1, Roberts in view of Chen fails to teach or suggest **linking said address to a pre-generated message interface for accessing said service, wherein said pre-generated message interface is implemented by computer-executable code installed on the device during a code-build process for installation of code on the device to implement said receiving and linking**.

Firstly, the Examiner's rejection fails to take into account the current, complete language of Applicants' claim. For instance, the Examiner argues and relies on Robert in view of Chen to teach "where a pre-generated message interface was constructed prior to runtime" (Office Action, pp. 6 and 11). However, Applicants claim does not recite "where a pre-generated message interface was constructed prior to runtime." Instead, as noted above, Applicants' claim recites, in part, wherein said pre-generated message interface is implemented by computer-executable code installed on the device during a code-build process for installation of code on the device to implement said receiving and linking.

In fact, the Examiner appears to have merely repeated an earlier rejection without having considered Applicants recent amendment at all (e.g., compare Final Action dated January 26, 2007, pp. 6-8 with current Action dated November 15, 2007, pp. 6-7). Nor has the Examiner address Applicants' recent arguments. Instead, as with the rejection of claim 1, the Examiner has simply repeated the same responses regarding claim 1 as in the January 26, 2007 office action.

As described previously, Roberts teaches that the interfaces relied upon by the Examiner are downloaded from a web services engine and used by a client application during runtime (see, Roberts, column 4, line 60 – column 5, line 7, column 5, lines 19-30, and column 7, lines 10-33). Specifically, Roberts teaches a runtime model that "draws on the use of a number of web services that construct special user interface pages as output data" and that "[t]he behavior of these pages is to generate subsequent web service requests to the web services engine, and to call for the execution [of] action defined in a web services application session" (Roberts, column 4, line 67 – column 5, line 5).

However, claim 1 requires that the pre-generated message interface is implemented by computer-executable code that was *installed during a code-build process for installation of code that implements the receiving and linking of the address* for the service to the pre-generated message interface. Thus, according to claim 1, the code that implements the pre-generated message interface is installed on the device during a same code-build process for installation (on the same device) of code that implements the receiving and linking. In contrast, Roberts describes a web services request that "generates a response [in] the form of a graphical user interface to be displayed in a browser". Roberts WSA interfaces are clearly meant to be downloaded and constructed *during runtime* and thus **Roberts teaches away** from the use of computer-executable code installed on the device during a code-build process for installation of code on the device to implement receiving an address for a service and linking the address to the pre-generated message interface. In response, the Examiner merely repeats the assertion, "Roberts disclosed where a pre-generated message interface was constructed prior to runtime" (emphasis by the Examiner, Office Action, p. 11), citing col. 13, lines 15-20). However, as noted above, claim does not does recite merely a pre-generated message interface constructed prior to runtime. The Examiner's assertion fails to address Applicants' specific argument. Even if Robert's system can be considered to include a pre-generated message interface constructed prior to runtime, that fails to disclose pre-generated message interface is implemented by computer-executable code that was *installed during a code-build process for installation of code that implements the receiving and linking of the address* for the service to the pre-generated message interface.

The Examiner also relies on Chen to teach use of dynamic agents to provide dynamic behavior modification of agents. Chen's agents are designed to carry application specific actions, which can be loaded and modified on the fly (e.g. at runtime). Chen's dynamic behavior modification allows a dynamic agent to adjust its capability for accommodating environmental and requirement changes. Chen's dynamic agents may also play different roles across multiple applications. The Examiner cites paragraph [0063] and refers to Chen's built-in APIs. Chen teaches built-in APIs that allows an action to create "a receiver thread and [to] register[] its socket address" (Chen, paragraph [0063]). APIs for creating threads and registering a socket address as taught by Chen even if combined with Roberts, do not teach or suggest a pre-generated message interface implemented by computer-executable code installed on the device during a code-build process for installation of code on the device to implement receiving and linking the address for the service to the pre-generated message interface.

Applicants note that the Examiner has failed to respond to the above argument as presented previously.

Thus, even if combined as suggested by the Examiner, Roberts and Chen fail to teach or suggest the specific limitations of claim 1. Instead, the combination of Roberts and Chen, as suggested by the Examiner would result in a system using the WSA interfaces of Roberts but that also includes built in APIs for creating receiver threads and registering socket addresses.

In the Response to Arguments section of the Final Action, the Examiner disagrees with Applicants' statement, "Roberts WSA interfaces are clearly meant to be downloaded and constructed *at runtime.*" The Examiner contends that Roberts' "templates build the program prior to running", citing column 13, lines 15-20. However, the Examiner is overlooking the specific language recited in claim 1. Claim 1 does not just recite that a pre-generated message interface is constructed prior to <u>running</u>, but that the pre-generated message interface is implemented by computer-executable code <u>installed on the device during a code-build process for installation of code on the device to implement receiving and linking the address for the service to the pre-generated message interface.</u>

Roberts' WSA interfaces are clearly not implemented by code installed on the device during a code-build process, as recited in claim 1. Instead, Roberts' WSA interfaces are "child runtime models" <u>dynamically constructed at runtime</u> according to lists of features contained in templates (see, e.g., Roberts, column 10, lines 14-25).

In the Response to Arguments, the Examiner contends, "at the time of the invention distributed object code including code generators installed on client devices were well-known in the networking art." Applicants respectfully traverse the Examiner's taking of official notice. Applicants assert that while "distributed object code including code generators installed on client devices" may have been well-known in other contexts, it was not well known to link an address for a service to a pre-generated message interface for accessing said service, where the pre-generated message interface is implemented by computer-executable code installed on the device during a code-build process for installation of code on the device to implement said receiving and said linking. Furthermore, pursuant to M.P.E.P. § 2144.03 Applicant asserts that "the examiner must provide documentary in the next Office action if the rejection is to be maintained. See also 37 CFR 1.104(c)(2), (d)(2) and *In re Zurko*, 258 F.3d 1379, 1386 (Fed. Cir. 2001).

Moreover, the Examiner has misunderstood Applicants' argument. Applicants are not arguing that code generators where not known. Applicants are arguing that Roberts WSA interfaces, even in view of Chen, are not implemented by code installed on the device during a code-build process *as recited in Applicants' claim*. In other words, Roberts WSA interface, relied on by the Examiner, even if combined with Chen, do not teach or suggest a pre-generated message interface implemented by computer-executable code <u>installed on the device</u> <u>during a code-build process for installation of code on the device to implement receiving and linking the address for the service to the pre-generated message interface</u>.

For at least the reasons above, the rejection of claim 1 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 14 and 27.

Regarding claim 8, the Examiner has again failed to provide a proper rejection under § 103 of this claim. Even through Applicants have repeatedly pointed this out to the Examiner, the Examiner again lists claim 8 as rejected under 35 U.S.C. 103(a), but **fails to provide an actual rejection**. Claim 8 recites subject matter different from that recited by claims 1-4, the only claims for which the Examiner provides rejections under 35 U.S.C. 103(a). Therefore, the rejection of claim 8 under § 103 is improper.

Further regarding claim 8, and as noted in Applicants' previous Response, **Roberts in view of Chen fails to teach or suggest** ***pre-generating at least one message interface for a device in order to access a service*** **by** ***receiving a schema defining messages for accessing the service; generating message endpoint code according to***

said schema; and linking said message endpoint code into executable operating code for the device and installing the message endpoint code and operating code onto the device.

In the Response to Arguments section of the Final Action, the Examiner disagrees with Applicants' previous assertion that Roberts and Chen do not teach linking message endpoint code, generated *according to a schema* defining messages for accessing a service, *into executable operating code* for a device. The Examiner cites Roberts (column 7, lines 10-15 and lines 45-50) as teaching a regeneration process for a transformed runtime model and fully interactive user interfaces, and that the runtime models follow a schema.

Applicants again assert that this general reference to "a schema" does not teach the limitations of claim 8, which recite that message endpoint code (i.e., code for a pre-defined message interface) is generated <u>according to a schema defining messages for accessing a service</u>. Applicants assert that "a schema" for a runtime model may specify many different things, and that it <u>may or may not have anything to do with *defining messages for accessing a service*</u>. Nothing in Roberts discloses that this particular type of schema is used in generating the child runtime models.

In the Response to Arguments, the Examiner contends, "Roberts disclosed constructing XML runtime models … for use with web services" citing col. 16, lines 20-25 and col. 17, lines 1-5 of Roberts (Office Action, p 12). However, the cited passage describe Roberts regeneration service and do not mention anything about message endpoint code generated according to a scheme defining messages for accessing a service. Instead, Roberts teaches "a set of features may be created for building XML-based messages." However, the Examiner appears to be overlooking the fact that Applicants claim requires more than simply "a set of features … for building XML-based messages." Building XML-based messages does not necessarily include receiving a schema defining messages for accessing a service and clearly does not necessarily include generating message endpoint code according to the schema, as recited in

Applicants' claim. The Examiner is apparently confusing the general creation of XML-based messages with the specific limitation of generating *message endpoint code* according a scheme defining messages for accessing the service. Merely generating XML-based message does not teach or suggest anything about how message endpoint code may be generated.

In addition, the regeneration process of building a runtime child model does not teach or suggest pre-generating at least one message interface for a device in order to access a service, as recited in claim 8. Roberts' user interfaces are described as "a set of dynamically generated and updated XML entities called Pages" in column 7, lines 11-13 (emphasis added.) Therefore, these child runtime models and user interfaces are clearly not pre-generated message interfaces that are generated and linked into executable operating code for a device, according to the limitations of claim 8. In response the Examiner argues that "templates represented by a block of XML data … are … equivalent to pre-generated message interfaces" citing col. 12, lines 55-60 of Roberts. However, the Examiner's interpretation is incorrect.

The Examiner cited passage fails to support the Examiner's conclusion. For instance, the cited passage does not mention anything about templates or blocks of XML being equivalent to pre-generated message interfaces. Instead, the cited passage clearly states that a template is represented by a block of XML data "that defines a set of input parameter values for the feature." Roberts fails to mention, either at the cited passage or anywhere else, anything about templates or blocks of XML data being equivalent to pre-generated message interfaces, as the Examiner contends.

Additionally, Chen, whether considered singly or in combination with Roberts, also fails to teach or suggest generating message endpoint code according to a schema defining messages for accessing a service, linking the message endpoint code into executable operating code for the device and installing the message endpoint code and operating code onto the device. The Examiner has not relied upon Chen or cited any portion of Chen regarding this limitation of claim 8. Thus, the

Examiner's combination of Roberts and Chen clearly fails to teach or suggest the limitations of claim 8.

Furthermore, as noted in Applicants' previous Response, Roberts specifically teaches downloading WSA interfaces from web service devices for use on client devices to access WSAs. Downloading of message interface is clearly quite different from linking message endpoint code, generated according to a schema defining messages for accessing a service, *into executable operating code* for a device and installing the message endpoint code and operating code onto the device. Thus, **Roberts teaches away** from Applicants' claim.

For at least the reasons above, the rejection of claim 8 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 21 and 28.

Regarding both the §102 and §103 rejections above, Applicants also assert that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-66200/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: February 15, 2008